PSTAT 5LS Lab 5

Professor Miller

Week of May 15, 2023

Section 1

Learning Objectives

R Learning Objectives

1. Revisit simulation for one proportion
2. Revisit `pnorm()` and `qnorm()`
3. Introduce `prop_test()` to understand test statistics and confidence intervals

Statistical Learning Objectives

1. Understand how area under the normal curve relates to probability
2. Understand how to move between probabilities and quantiles of the normal distribution
3. Build intuition for the relationship between simulated p-values and p-values which arise from a normal approximation.
4. Understand the standard normal distribution and its corresponding z test statistic.
5. Understand how confidence intervals can provide an estimate for the true parameter.

Functions covered in this lab

1. `plot_norm()`
2. `pnorm()`
3. `qnorm()`

Section 2

Lab Tutorial

Another Simulation Example

According to the American Pet Products Association, prior to the pandemic, 67% of all U.S. households had pets. Many people have speculated that the rate of pet ownership changed during the pandemic. Recently, a group of veterinarians surveyed a random sample of 480 U.S. households and found that 336 had at least one pet.

Is there evidence to support the claim that the rate of pet ownership now differs from the 67% before the pandemic?

Our hypotheses to test if there is a difference are

$$H_0 : p = 0.67 \text{ and } H_A : p \neq 0.67$$

Setting Up the Simulation

	Assuming the chance model...
One draw	*one household*
Blue poker chip	*the household has at least one pet*
Yellow poker chip	*the household has no pets*
Chance of blue	*0.67*
One repetition	*480 households*

Running the Simulation

Recall that we need to set a seed before running a simulation. Let's arbitrarily set the seed to 734. Run this in the `setSeed` code chunk.

Then let's run the simulation 500 times.

```
sim1 <- simulate_chance_model(chanceSuccess = 0.67,
                              numDraws = 480,
                              numRepetitions = 500)
```

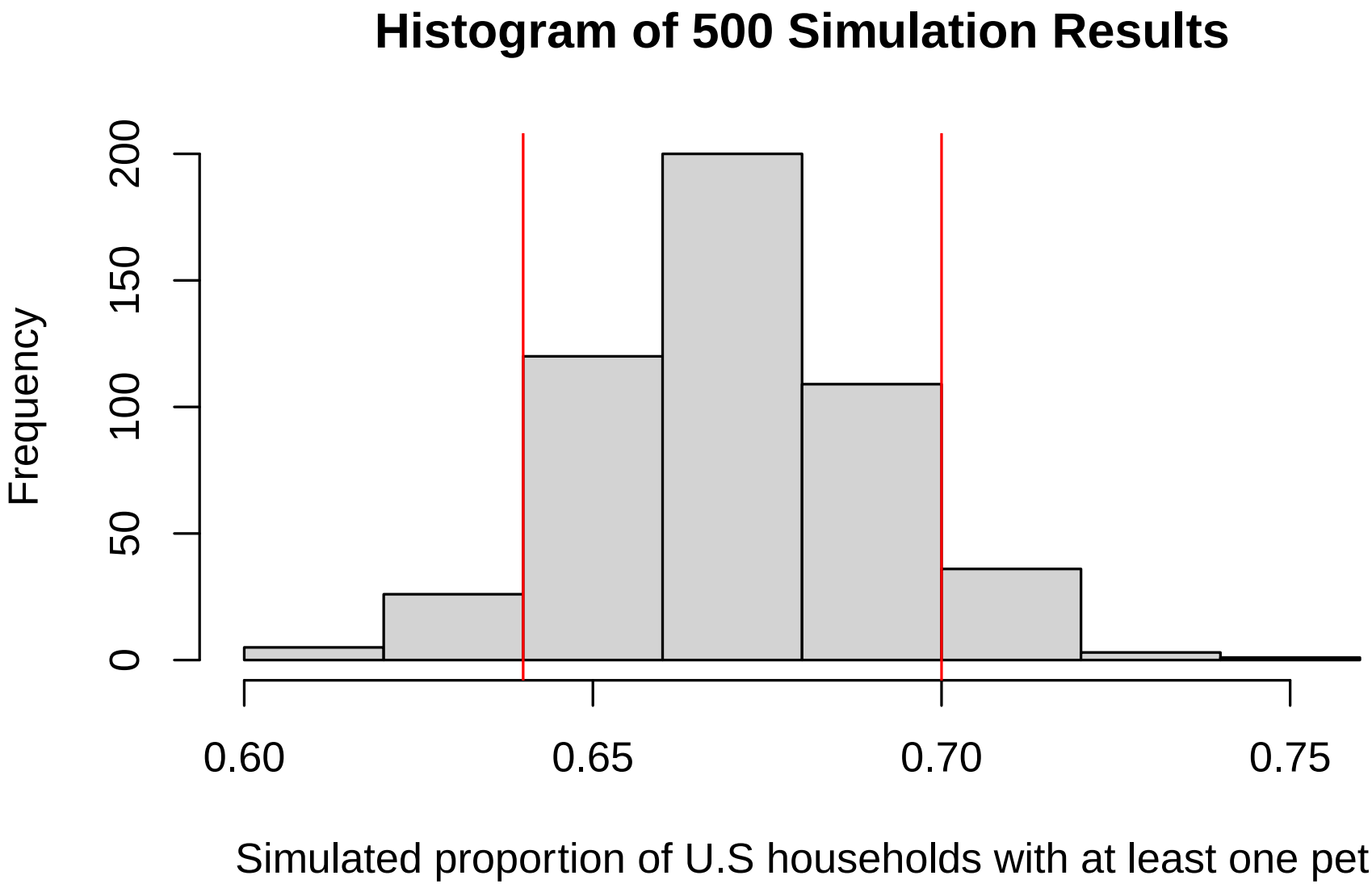
`sim1` should now be in your Environment pane.

Histogram of Simulation Results

The sample statistic is $\hat{p} = \frac{336}{480} = 0.70$, so we will put a line in the histogram to indicate our observed statistic.

In our two-sided hypothesis test, it would have been just as unusual to see a sample proportion of 0.64, so we will put a line in the histogram to indicate this as well.

Histogram of Simulation Results



Estimated p-value

The estimated p-value is the sum of the simulations that are as weird as or weirder than we observed (so, 0.70 or larger) *and* the sum of the simulations that are as weird or weirder on the other side (at 0.64 or less).

```
## [1] 0.156
```

Our estimated p-value is 0.156.

Conclusion

After running the simulation 500 times, assuming that the chance of success was indeed 0.67, the probability that we would get an observed sample proportion of 336/480 or more extreme was computed to be 0.156.

The p-value of 0.156 is larger than even a significance level of 0.10. We have little to no evidence to support the claim that the rate of pet ownership now differs from the 67% before the pandemic.

But We Observed a Different Result?!

You might have asked yourself, "But wait, why do we have little to no evidence to support the claim? I did in fact observe a rate different than 67% in my sample (I observed 70%, in fact). Isn't a 3% difference strong evidence?"

This is where the normal theory can help us understand that the difference of 3% between the $\hat{p}$ and the assumed p was not enough evidence.

Reminder about Normal Distributions

Recall that a "distribution" refers to the possible values a random variable can take as well as the probability that it takes those values. It is commonly used to approximate all sorts of things in nature and life.

It takes two numbers to completely describe a normal distribution: the **mean** and **standard deviation**. We denote a normal distribution by N(μ, σ), where μ is the population mean and σ is the population standard deviation.

Normal distributions are all bell-shaped, unimodal, and symmetric about their means, regardless of the values of the mean (μ) and the standard deviation (σ).

The mean μ specifies the *center* of the distribution. The standard deviation σ specifies the *variability* of the distribution (meaning, how narrow or wide is it?).

Histogram from Simulation

It turns out, the histogram that we created using simulation is in fact approximately normal.

We talked about how the distribution is centered at H_0, the chance of success. Thus we can assume $\mu = 0.67$ in our example about pet ownership prior to the pandemic.

To find the standard error, which is an estimate of σ, we will use the following formula:

$$\sqrt{\frac{p_0(1-p_0)}{n}}$$

For the example about pet ownership, let's compute the standard error, $SE(\hat{p})$.

Histogram from Simulation

$$\sqrt{\frac{p_0(1-p_0)}{n}} = \sqrt{\frac{0.67(1-0.67)}{480}} = 0.021$$

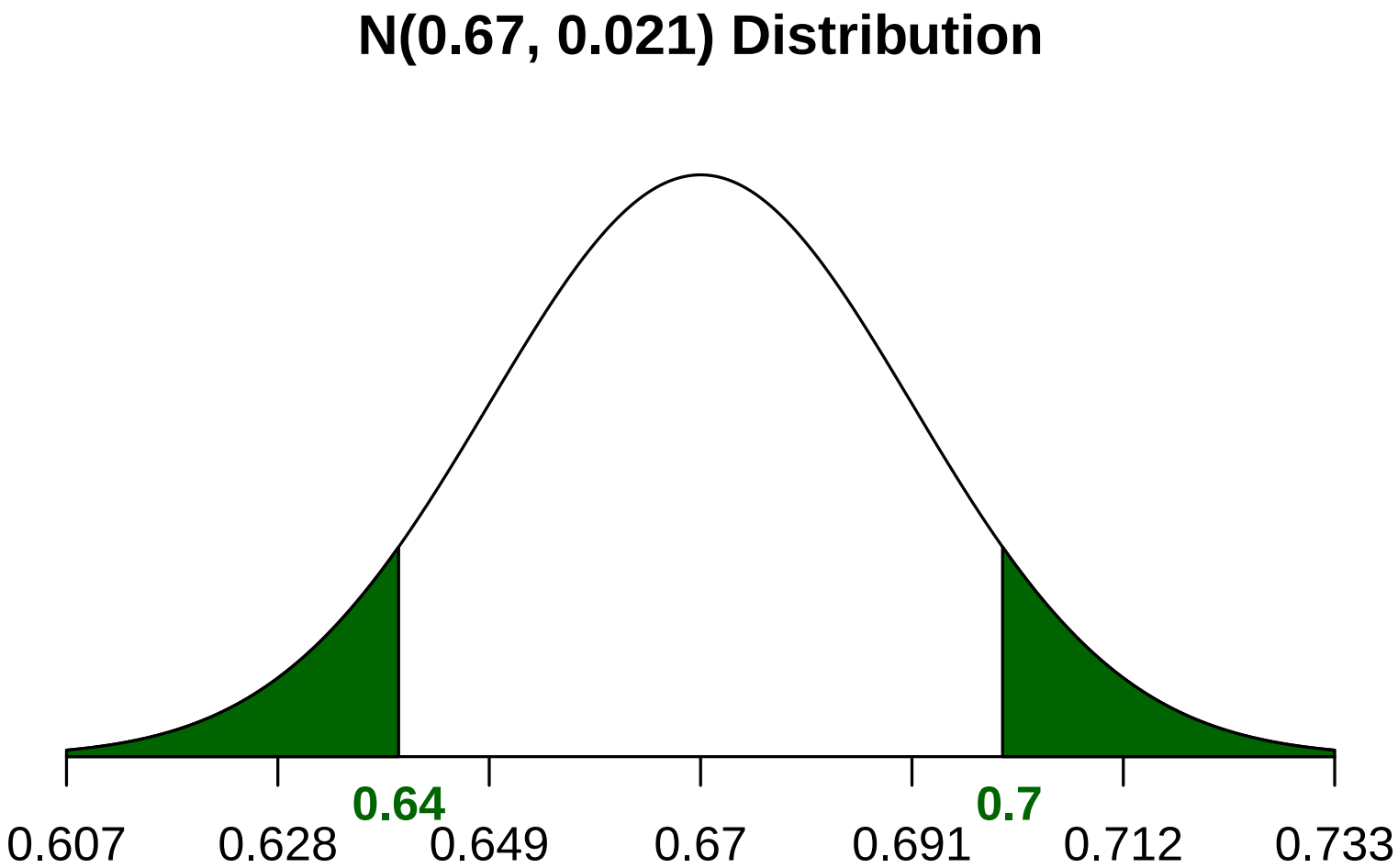
Thus, the example about pet ownership is approximately N(0.67, 0.021).

Normal Distribution for Our Example

Let's make the approximate normal distribution for our pet ownership example. Recall that, when the null hypothesis is true, the mean is **0.67**, the standard error is **0.021**, and we want to shade **beyond** the values of 0.64 and 0.7. Try this code out in your notes.

```
plot_norm(mean = 0.67,
          sd = 0.021,
          shadeValues = c(0.64, 0.7),
          direction = "beyond",
          col.shade = "darkgreen")
```

Normal Distribution for Example



There Is in Fact Insufficient Evidence

Now that we see that the standard deviation is 0.021, or 2.1%, we can start to understand why observing a rate that was only 3% different from the assumed mean is not all that rare.

Looking at the normal distribution plot, we can observe that 0.7 is only between 1 and 2 standard deviations above the mean. Again, not all that rare.

Another Way to Calculate the p-value

The `stats250sbi` package we have been using also contains a function called `prop_test()` that will calculate the value of the test statistic and the p-value for us.

You will need to send the function the following arguments:

- `x`: the number of "successes" in the sample
- `n`: the sample size
- `p`: the hypothesized population proportion
- `alternative`: where to shade (`"two.sided"`, `"less"`, `"greater"`)
- `conf.level` (optional): used to get a confidence interval (we will use this later)

Using prop_test for Our Pet Ownership Example

The code we need for the test of $H_0 : p = 0.67$ and $H_A : p \neq 0.67$ is

```
prop_test(x = 336,
          n = 480,
          p = 0.67,
          alternative = "two.sided")
```

Using prop_test for Our Pet Ownership Example

```
##
##  1-sample proportions test without continuity correction
##
## data:  x out of n, null probability p
## Z = 1.3978, p-value = 0.1622
## alternative hypothesis: true p is not equal to 0.67
## 95 percent confidence interval:
##  0.6590044 0.7409956
## sample estimates:
##   p
## 0.7
```

The pnorm() Function

The p-value from `prop_test()` is pretty close to the p-value from our simulation.

We can also compute the p-value using the `pnorm()` function. Recall the arguments you need to send to `pnorm()`:

- `q`: the quantile (value on the axis) for the normal distribution
- `mean`: the mean of the normal distribution (μ)
- `sd`: the standard deviation of the normal distribution (σ)
- `lower.tail`: set to 'TRUE' initially, signifying that R will compute the probability **to the LEFT** of `q`; if you would like R to compute the probability *to the right* of `q`, set `lower.tail` to FALSE

Computing the p-value for the Simulation Example

Let's compute the approximate p-value using the normal distribution for our pet ownership example. Recall that the mean is **0.67**, and the standard deviation is **0.021**.

Because the normal distribution is **symmetric** about the mean, we can find the probability of observing 0.64 or less, then **double it**.

Or, we can find the probability of observing 0.7 or greater, then **double it**.

Or, we can find the probability of observing 0.64 or less, then the probability of observing 0.7 or greater, and add the result.

Since 0.7 is the $\hat{p}$ for our sample, we will utilize the second option in the `pvalue` code chunk.

Computing the p-value for the Simulation Example

Try this code out in the `pvalue` code chunk.

```
2 * pnorm(q = 0.7,
      mean = 0.67,
      sd = 0.021,
      lower.tail = FALSE)
```

```
## [1] 0.1531275
```

Comparing the Various p-values

To recap, we have *three* ways to compute the p-value for a one proportion hypothesis test:

1. Create a vector of simulated proportions using `simulate_chance_model()`, then using the `sum()` function to count the number of observations at or beyond the sample proportion divided by the number of observations
2. Use the `prop_test()` function (which uses the normal theory) by sending the number of successes observed, the sample size, the value of H_0, and the direction of the alternative hypothesis
3. Compute the μ and σ for the approximate normal distribution. Use the `pnorm()` function by sending the value of the sample proportion, μ, σ, and the direction of the probability

Each of these will produce a slightly different result. **No need to worry about how close the values should be, or which value is "best".**

Comparing the Various p-values

```
sum(sim1 <= 0.64) / 500 + sum(sim1 >= 0.7) / 500
```

```
## [1] 0.156
```

```
prop_test(x = 336, n = 480, p = 0.67)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  x out of n, null probability p
## Z = 1.3978, p-value = 0.1622
## alternative hypothesis: true p is not equal to 0.67
## 95 percent confidence interval:
##  0.6590044 0.7409956
## sample estimates:
##   p
## 0.7
```

```
pnorm(q = 0.7, mean = 0.67, sd = 0.021, lower.tail = FALSE) * 2
```

```
## [1] 0.1531275
```

Comparing the Various p-values

1. The simulation is the most accurate, because it is computing the p-value with simulated values.
2. `prop_test()` and `pnorm()` will lose some precision due to utilizing the normal approximation. This loss of precision should not affect our results.
3. `pnorm()` will lose some precision if we round the standard deviation to 3 decimal places. This loss of precision should not affect our results.

Using prop_test() to Find Confidence Intervals

The output from prop_test() also provides a confidence interval for the population proportion. The default confidence level is 0.95 for a 95% confidence interval. The 95% confidence interval for this example is (0.659, 0.741).

```
##
##  1-sample proportions test without continuity correction
##
## data:  x out of n, null probability p
## Z = 1.3978, p-value = 0.1622
## alternative hypothesis: true p is not equal to 0.67
## 95 percent confidence interval:
##  0.6590044 0.7409956
## sample estimates:
##   p
## 0.7
```

Using prop_test() to Find Confidence Intervals

We can change the confidence level by adding the argument `conf.level` to the `prop_test()` function. For example, a 98% confidence interval is (0.651, 0.749).

```
##
##  1-sample proportions test without continuity correction
##
## data:  x out of n, null probability p
## Z = 1.3978, p-value = 0.1622
## alternative hypothesis: true p is not equal to 0.67
## 98 percent confidence interval:
##  0.6513409 0.7486591
## sample estimates:
##   p
## 0.7
```

Using `prop_test()` to Find Confidence Intervals

Caution: To get a two-sided confidence interval, the `alternative` argument *must* be set to `two.sided`. If it isn't, you will get a *confidence bound*.

Confidence bounds can be useful when we have one-sided hypothesis tests, but we will leave them to your later statistics courses.